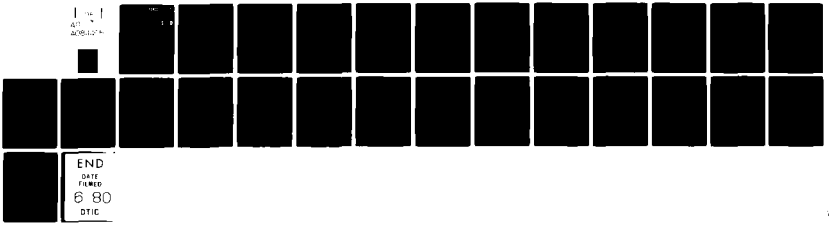
AD-A084 405 MICHIGAN STATE UNIV EAST LANSING DEPT OF COMPUTER SCIENCE F/G 9/2
A HIGHLY PARAMETERIZED TOOL FOR STUDYING PERFORMANCE OF COMPUTE--ETC(U)
MAR 80 H D HUGHES AFOSR-78-3547

UNCLASSIFIED AFOSR -TR-80-0339 NL



END
DATE FILMED
6 80
DTIC

(12)

(6) A HIGHLY PARAMETERIZED TOOL FOR STUDYING PERFORMANCE OF COMPUTER SYSTEMS*

(9) Interim rept.

(10) Herman D. Hughes
Department of Computer Science
Michigan State University
East Lansing, Michigan 48824

(11) Mar 80

(12) 27

ABSTRACT--A highly parameterized simulation model is described which allows experiments to be performed for computer performance evaluation studies. The results of these experiments can be used to evaluate the effect of changing the hardware configuration, the workload, the scheduling policy, the multiprogramming level, etc. The model is constructed to function either as a batch or time-sharing system, or as a combination of both. This simulation model also has the potential of providing dynamic feedback for the scheduler. A discussion of the design, implementation, and use of the model is presented. Examples are provided to illustrate some possible uses of the model and verifications of the results obtained from the model.

(14) (15) AFOSR-78-3547

KEYWORDS--Simulation model, queue, scheduling policies, workloads, hardware configuration, model validation, system performance, events, cumulative distribution function.

*Research supported by AFOSR Grant 78-3547

411 724

I. INTRODUCTION

In order to provide sufficient information for evaluating changes to computer systems, both the hardware and software must be evaluated with respect to the efficiency in performing their required tasks. There are certain realistic constraints which make it virtually impossible to effect changes to existing systems for the purpose of studying computer system performance. Many of these constraints, however, may be overcome by the use of a flexible computer simulation model [2].

An emphasis of this investigation is to focus on providing a tool for assisting analysts in making decisions on different scheduling strategies. In order to develop such a tool, it is obvious to this investigator that a fairly complex model of a computer system is required.

This paper describes a highly parameterized simulation model which allows experiments to be performed for which the hardware configuration, the workload, and the scheduling policy can vary. The model is event-driven and is designed to accommodate systems as simple as batch with uniprogramming, to more complex systems which make use of time-sharing, multiprogramming, and virtual memory principles. Major components of the model are described in the next section of this paper.

Several experiments are presented to illustrate the potential use of the simulation model. Typical output from the model includes: performance indicies (i.e., response time, throughput rate, dilation, paging rate, swapping rate, etc.), queue statistics, utilization measures, and a profile of the system. This output is available at the job-step level or at the overall system level, and is broken down by system overhead and users' statistics.

This model may serve as a tool for providing guidance to system analysts, capacity planners, and individuals involved in courses such as system programming, operating systems, simulation, and performance measurements/evaluations.

II. DESCRIPTION OF MODEL

The model is written in a high-level language--ANSI standard FORTRAN--and is implemented on a CDC Cyber 750 computer. There are several components to the model, and each component corresponds to a FORTRAN subroutine. These components and their functions will be examined after a discussion of the flow of transactions through the system.

The high-level flow of jobs (job-steps) through the system is dipicted in Figure 1. Each job-processing step listed below corresponds to an event within the model 7 .

Step 1:

A job (batch or interactive) arrives randomly or according to a specified distribution. Upon arrival, the following job characteristics are determined either randomly or according to a pre-defined distribution: (1) the total CPU time, (2) the average amount of central memory (CM) requested, and (3) the number of I/O requests.

Step 2:

The job makes a request for CM allocation. If the CM space requested is not available, the job enters the CM queue.

Step 3:

After the job enters the CM, it immediately requests the CPU. If the CPU is free, it is assigned to the job and executes until some blocking condition occurs (i.e., a system interrupt, the time-slice used up, the job is completed, or an I/O request is encountered). In the former two cases, the job releases the CPU, but is placed back into the CPU queue.

Step 4:

When a job issues an I/O request, the CPU is released, and a specific disk is requested. Since the total CPU time and the number of disk requests for a job are predetermined, it is assumed that the inter-I/O request time is constant for a given job.

Step 5:

In order for a job to access a designated disk, both the disk and the associated channel must be free. Otherwise, the job enters a disk queue. If the disk and the channel

are both free, a "disk-seek" time is generated. During the "disk-seek" time, the disk is busy, whereas the channel is not.

Step 6:

After completing the disk-seek, a "rotational delay" time is generated. When this time expires, the channel is requested again, and if available, the data is transferred over the channel. The disk and the channel are both busy during the "transfer" time.

Step 7:

When the data transfer is completed, the disk and the channel are both freed, and the job proceeds to request the CPU again.

Step 8:

Upon completing all the CPU and I/O tasks for a given job, the CM allocated for that job is released. If the job is a batch job, it leaves the system; otherwise, the job is an interactive job, and has just completed a "system response cycle", so a "user think-time" is generated.

III. JOB EVENTS

The job-processing steps listed by Steps 1-8 represent only a subset of the events within the model. Other events included in the model are highlighted by Figure 2. Those events which appear in the flowchart boxes have event-times which are predetermined and, therefore, can be placed on the future-event list. The set of events whose event-times cannot be determined in advance are just listed below the flowchart (refer to Figure 2). For example, if a batch job is in the CM-queue, the next event is requesting CM; but since it depends on when other jobs will leave the system and make space available, the event time cannot be determined. On the other hand, if a job seizes the CPU at time t_o, and the inter-I/O request time T is known, then the next event for this job (release the CPU) can be scheduled at time t_o+T, and hence placed on the future-event list.

IV. QUEUE STRUCTURE FOR MODEL

The model consists of several queues (i.e., future-event queue, CM queue, CPU queue, channel queue, free-record queue, disk queue, etc.). Each of these queues forms a

ring with a coincident head and tail. Records in the queues are constructed as doubly-linked lists with pointers to the immediate predecessors and successors.

Job-records in the future-event queue are always in ascending order of the next event-time, whereas jobs in each waiting queue are always in decending order of job priority.

Figure 3 illustrates a typical queue structure for job-records within the model. Notice that a job (job-record) can only appear in one of the queues at a time, and that records which are not currently active are attached to the free-record queue.

The average queue length ($\bar{\ell}$ n) can be derived as follows:

$$\bar{\ell}_n = [\ell_o(t_1-t_o)+\ell_1(t_2-t_1)+\ldots+\ell_{n-1}(t_n-t_{n-1})]/(t_n-t_o)$$

$$=[-\ell_o t_o+(\ell_o-\ell_1)t_1+(\ell_1-\ell_2)t_2+\ldots+(\ell_{n-1}-\ell_n)t_n+\ell_n t_n]/(t_n-t_o)$$

$$=[\sum_{i=1}^{n}(\ell_{i-1}-\ell_i)t_i+\ell_n t_n]/t_n$$

$$\bar{\ell}_n = [\sum_{i=1}^{n}(\ell_{i-1}-\ell_i)t_i]/t_n+\ell_n$$

where $\{t_i\}_o^n$ denotes the time when a job enters or leaves the queue, and ℓ_i is the queue length at time t_i, with t_o=o. This formula was used throughout the model for calculating average queue lengths.

V. THE GENERATION OF DISTRIBUTIONS

Distributions used by the model are generated via a set of Cumulative Distribution Functions (C.D.F.s). These C.D.F.s are defined by the user supplying a set of discrete functions (e.g., 11 points), thus permitting the generation of a desired distribution. As an example, assume that the job arrives according to Poisson distribution with mean $\lambda = 3$ jobs/sec. Then, the inter-arrival time (random variable X) is known to have an exponential distribution of mean $\theta = 1/3$, hence the C.D.F. function can be approximated as follows:

$$F[X \le t] = 1 - e^{t/\theta} = 1 - e^{3t}, \text{ where } F[X \le t]$$

takes on the values 0.0, 0.1, 0.2, ..., 0.9, 1.0 :

F [X ≤ t]	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	0.9995*
t(approx.)	0	0.0351	0.0744	0.1189	0.1703	0.2310	0.3054	0.4013	0.5365	0.7675	2.534

*Here, since F [X ≤ t] equals 1.0 only when $t \to \infty$, we use the value 0.9995 in order to avoid $t = \infty$. The approximated C.D.F. is shown in Figure 4(a).

After the C.D.F. has been approximated, and a sample is desired from this distribution, we only need to generate uniformly distributed random numbers over the unit interval (0,1), and perform an inverse transformation on the C.D.F.. This transformation involves a table look-up and a linear interpolation procedure to obtain sample values. A possible pitfall of this approach is that when a discrete function is used to approximate a continuous function, some error must be tolerated. Figure 4(b) illustrates the case where a value x' may be generated which is slightly larger than the actual value x. Clearly, as the number of points used to approximate a function increases, the accuracy of the sample values also increases.

Another problem related to the generation of various distributions in modelling is the independence of the set of random numbers generated for different distributions. For this model, the independence of the generation of jobs within job-classes was achieved by using a different random-number seed for generating each job-class.

VI. MODELLING VARIOUS COMPONENTS OF SYSTEM

HARDWARE (see Figure 1) - The following hardware is modelled, and can be configured in various ways.

CPUs

DISKS

CHANNELS

TERMINALS

MEMORY

The timings of the hardware components are relative, and may be redefined by the user.

SOFTWARE - In order to simplify the model, the details of the operating system are not modelled explicitly; instead, timings for system overhead are included in system tasks (i.e., paging, swapping, scheduling, etc.) [1] .

PAGING - Paging is modelled as a high-priority system job which is activated at a certain rate (specified as a parameter by the user). This high-priority job will use the CPU, channel, and the disk to read/write one page of data. By using this approach for modelling paging, the contention for devices can be simulated very easily.

The paging rate is defined for some fixed multi-programming level (i.e., MPL=7), and will vary as a function of: (1) the number of interactive users, (2) the memory size, and (3) the MPL level.

SWAPPING - Swapping is not modelled explictly; instead it is modelled as a high-priority system job which is activated when (1) TTY jobs get into or out of think-time (CM is actually freed), or (2) jobs request disk I/O, but are blocked. An input parameter controls the swapping rate. If the CM-queue length is greater than this input parameter, swapping occurs.

The system resources used by swapping are: the CPU, disk, and the channel.

STORAGE ALLOCATION - the acquisition and release of main storage for the application programs are modelled. The user specifies the memory size via input parameter.

SCHEDULING - Originally, jobs coming from the same class (batch, system, or interactive) are assigned the same priority (specified as a parameter). This convention may be altered if it is desired to assigned different priorities to jobs within the same class. Each time a job changes queues, its priority is recalculated. The calculation proceeds as follows:

Internal priority = original priority + (CPU time used) x $weight_1$

+ (system residence time) x $weight_2$ + (CM size) x $weight_3$

where $weight_i$ are input parameters.

By altering input parameters such as initial job priority, internal priority weight, quantum size, MPL, etc., different scheduling algorithms can be investigated. Since the model collects statistics such as queue lengths, and utilization information, the results of this statistics can be used to provide dynamic feedback to the scheduler [3].

WORKLOAD - Each batch job-class is characterized by its CM request, CPU time, and number of disk I/O requests. An interactive job is characterized by its CM request, CPU time, number of disk I/O requests, and the length of its think-time. These job characteristics are defined by distribution functions. For example, a user's think time may be simulated by sampling from an exponential distribution with mean = 16 [5]. An approach for generating representative workload data to drive a simulation model will be discussed in a forthcoming paper.

VIII. MODEL VALIDATION AND EXPERIMENTAL RESULTS

It is an established fact that the validation of a simulation model is a complex process [4]. This model was validated by (1) verifying the logic of the FORTRAN program, (2) using a constant model to verify the accuracy of the statistics produced by the simulation model, and (3) using stochastic processes to check the correctness of the simulation. Since the first two steps used for validation are straightforward, the results which make use of stochastic processes (step 3) are presented in experiments which follow.

Experiment 1.

no. of CPU = 1

no. of disk = 1

no. of channel = 1

size of CM = 300K words

multiprogramming level = 1 (uniprogramming)

The job-parameters were:

mean arrival rate λ =3 jobs/second (Poisson distributed) from a single batch class.

mean CPU service time μ_1=0.1 sec/job
mean disk service time μ_2=0.05 sec/job
avg. CM request per job = 50K words
} distributed exponentially.

The system is depicted in Figure 5. A job enters the system only when there is no other job running. It uses one half of the CPU time requested, visits the disk once, uses the second half of the CPU time, then leaves the system. It should be noted that the channel is not modelled here, since no contention exists for it in a uniprogramming mode.

The simulation results are tabulated in Table 1 along with the analytic results. An explanation for the calculations made in rows 1-6 of Table 1 is given below.

1. Row 1 -- λ is given as an input parameter. (λ =3)
2. Row 2 -- the CPU utilization ($\cup$ CPU) is calculated as:

   $\cup$ CPU = λ μ_1 = 3.0 x 0.1 = 0.3
3. Row 3 -- the disk utilization ($\cup$ disk) is calculated as:

   $\cup$ disk = λ μ_2 = 3.0 x 0.05 = 0.15
4. Row 4 -- the turnaround time R is calculated as:

$$R = E(\text{service time}) + \frac{\lambda\, E(\text{service time}^2)/2}{1 - \lambda\, E(\text{service time})}$$

$$= (0.15) + \frac{\frac{3}{2}}{1-3(0.15)} \left[(0.15)^2 - (905)(0.1)\right]$$

$$= 0.15 + 0.075 = 0.245 \text{ seconds}$$

5. Row 5 -- avg. CM queue length = total wait time/total time

   = (turnaround time - service time) x (no. of job completion)/total time

   = (0.245-0.15) x throughput $\simeq$ 0.095 x 3.0 = 0.285
6. Row 6 -- avg. CM utilization = E(service time) x E(CM request) x throughput/CM size

   $\simeq$ (0.15) x (50) x 3.0/300 = 7.5 x 3.0/300 = 0.075

Also, the Operational Approach proposed by Buzen [8] can be used to check the consistency of the model as follows:

Little's law states that the average number ($\bar{n}$) of jobs in the system, including those waiting in the CM-queue, is given by:

$\bar{n} = xR$ where x = throughput

R = turnaround time.

Hence, we have

$R = \bar{n}/x$

Now, $\bar{n}$ = avg. CM queue length + $\cup$ CPU + $\cup$ disk

= 0.184 + 0.296 + 0.149 = 0.629 in the 400-job case,

and $x \approx 2.48$, hence

$R = \bar{n}/x \approx 0.629 \times 2.48 = 0.255$ seconds, which is close to the result for the 400-job case.

Experiment 2.

To further validate the model, let's suppose that during a job's access to the disk, a disk-seek time and a latency (rotational delay) time were generated. Consider the following configuration:

no. of CPU = 1

no. of disk = 1

no. of channel = 1

size of CM = 300K words

multiprogramming level = 1 (uniprogramming)

The job-parameters were:

mean arrival rate $\lambda = \frac{1}{3}$ jobs/second (Poisson dsitributed) from a single batch class.

avg. disk-seek time = 0.04 seconds
avg. latency = 0.01 seconds
} distributed exponentially.

The system is depicted in Figure 6, and the results are shown in Table 2.

Considerably more validation was done on the simulation model, but will not be presented in this paper [4] . The remaining experiments are presented to illustrate some of the more interesting outputs from the model. Appendix I contains an example of a system profile produced by the model. A system profile enables one to observe the degree of overlap of resource utilization during a selected time interval.

Experiment 3.

This experiment is to investigate the effects of varying the quantum (time-slice) size and observe the performance of a multiprogramming computer system. The system under study has the following configuration:

no. of CPU = 1

no. of disks = 8

no. of channels = 2; 4 disks/channel

size of CM = 128K

multiprogramming level = 10

system overhead due to job-swapping = 2 to 3 msec.

The impact of various quantum sizes on the system's behavior is plotted in Figure 7. Basically, for quantum sizes of 1.0 to 0.3 seconds, the system performs much the same, because the average inter-I/O time is relatively small compared to the quantum sizes. As the quantum size decreases to 0.08 seconds, the turnaround time and the CPU queue length are considerably reduced, hence we get better performance from the system. However, if the size of the time-slice gets too small, the system overhead increases significantly, and therefore degrades the system performance. So, this simulation can provide some guidelines for determining the size of the time-slice.

Experiment 4.

In order to analyze the effects of different multiprogramming level (MPL) on the sytem performance, a set of experiments was performed with various MPLs. The general

configuration is depicted in Figure 1, with the following specifications:

no. of CPU = 1

no. of disks = 8

no. of channels = 2; 4 disk/channels

size of CM = 128K

quantum size = 0.1 second

disk-seek time = 0.04 seconds
rotational delay = 0.01
disk service time = 0.2 seconds
} distributed exponentially.

Batch jobs (jobsteps):

mean arrival rate λ = 1/2.8 jobs/second (Poisson distributed)

mean CPU service time = 2.0 seconds
avg. no. of disk I/O = 5 times
} distributed exponentially.

Interactive jobsteps:

no. of terminals = 10

user think-time, Z = 18 seconds
mean CPU service time = 0.2 seconds
avg. no. of disk I/O = 2 times
} distributed exponentially.

The system also has paging and swapping overhead as explained in previous sections.

Figure 8 shows the plot of the MPL vs system performance in terms of batch turnaround-time, TTY response time, system overall throughput, and system overhead, etc.

Experiment 5.

Suppose that the system is now dedicated to interactive users, and we wish to study the behavior of the response-time as the number of terminals increases. The system has the same configuration as described in Experiment 4, except that the MPL is set at 7. Workload characteristics for this experiment are described by the following parameters:

mean CPU service time per interaction = 0.2 seconds
avg. no. of disk I/O requests = 2 times
user think-time = 18 seconds
} distributed exponentially.

Figure 9 shows the various performance indices obtained as a result of varying the number of terminals.

VIII. SUMMARY AND CONCLUSION

We have presented a general-purpose simulation model which is capable of simulating a wide variety of computer systems. The major advantages of this model can be characterized as the following:

i. the structure of the model is general enough to be tailored for many computer systems, and yet,

ii. the model is highly parameterized so that it can closely approximate a real system by specifying the hardware and software configurations;

iii. the (batch and interactive) workloads that drive the simulation model can also be defined handily by a set of job-parameters;

iv. the model can be easily modified to accomodate different scheduling algorithms.

Several uses of the model may be cited. It can serve as a tool for the analysis of system performance due to upgrading or changing scheduling policies. It may also be used to predict the system's future performance with different workloads. Section VII illustrated some of these applications by a set of experiments. While the numerical results of the simulation model may not be completely accurate; it nevertheless indicates the trend of improvements or degradations, thereby providing guidelines to the analysis of complex computer systems.

ACKNOWLEDGEMENTS

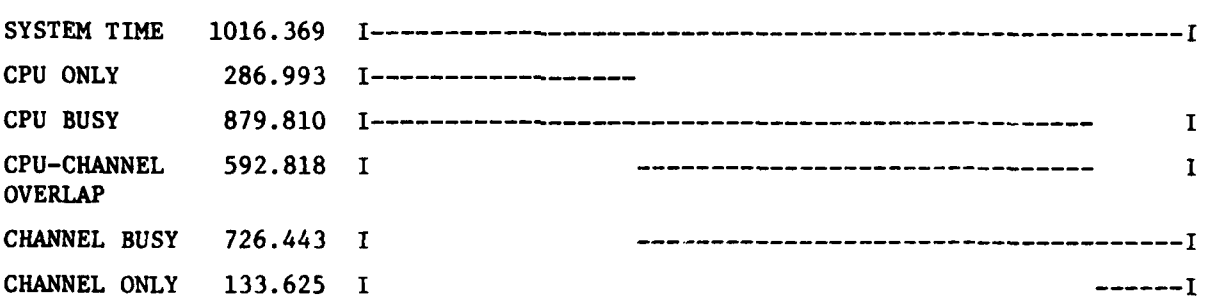
I would like to express my appreciation for two graduate students (Liang Li and Jeff Perdue) who provided a significant contribution to the development of this paper.

APPENDIX I

SYSTEM PROFILE FOR THE SIMULATION RUN:

CPU AND CHANNEL STATUS: 0=IDLE; 1=BUSY.

CPU	1			TIME FOR EACH COMBINATION
	0			136.579
	1			879.810
CHANNEL	1	2	3	TIME FOR EACH COMBINATION
	0	0	0	289.946
	0	0	1	70.455
	0	1	0	206.797
	0	1	1	40.659
	1	0	0	154.167
	1	0	1	31.632
	1	1	0	193.887
	1	1	1	28.627



REFERENCES:

1. Hughes, H.D., "GPSS Simulation Model of MVS", Interim Technical Report, Dow Chemical, September 1979.

2. Schwetman, H.D. Jr., A Study of Resource Utilization and Performance Evaluation of Large-Scale Computer Systems, Ph.D. thesis, the University of Texas at Austin, Computation Center, August 1970.

3. Bunt, R.B. and Hume, J.N.P., "A Simulation Study of A Demand-Driven Scheduling Algorithm", Symposium on Simulation of Computer Systems III, 1975.

4. Therey, Toby J., "Validation Criteria for Computer System Simulations", Symposium on Simulation of Computer Systems III, 1975.

5. Hughes, H.D., "Some Predicted Results for the APL System", Abstracted in the Proceedings of the ACM Computer Science Conference, 1978.

6. Coffman, E.G. Jr., and Wood, R.C., "Interarrival Statistics for Time-Sharing Systems", Communications of the ACM, Vol. 9, No. 3, July 1966.

7. McDougal, M.H., "Computer System Simulation", Computing Surveys, Vol. 2, No. 3, 1970.

8. Denning, P.J. and Buzen, J.P., "The Operational Analysis of Queuing Network Models", Computer Surveys, Vol. 10, No. 3, September 1978.

ITEM	Simulation results			Theoretical
	200 jobs	300 jobs	400 jobs	results
Mean arrival rate λ ($\simeq$ throughput)	2.52	2.68	2.48	3.00
Avg. CPU utilization	0.293	0.295	0.296	0.300
Avg. disk utilization	0.147	0.148	0.149	0.150
Turnaround time	0.242	0.232	0.255	0.245
Avg. CM queue length	0.166	0.172	0.184	0.285
Avg. CM utilization	0.090	0.083	0.085	0.075

TABLE 1. Simulation and Analytic Results for Experiment #1

ITEM	Simulation results for 300 jobs		Theoretical
	Seek & latency constant	seek & latency exponentially distributed	results
Mean arrival rate λ ($\simeq$ throughput)	0.296	0.296	0.333
Avg. CPU utilization	0.656	0.656	0.667
Avg. disk utilization	0.146	0.156	0.150
Avg. channel utilization	0.135	0.143	0.133
Turnaround time	4.684	4.732	4.689
Avg. CM queue length	0.574	0.578	0.658

TABLE 2. Simulation and Analytic Results for Experiment #2

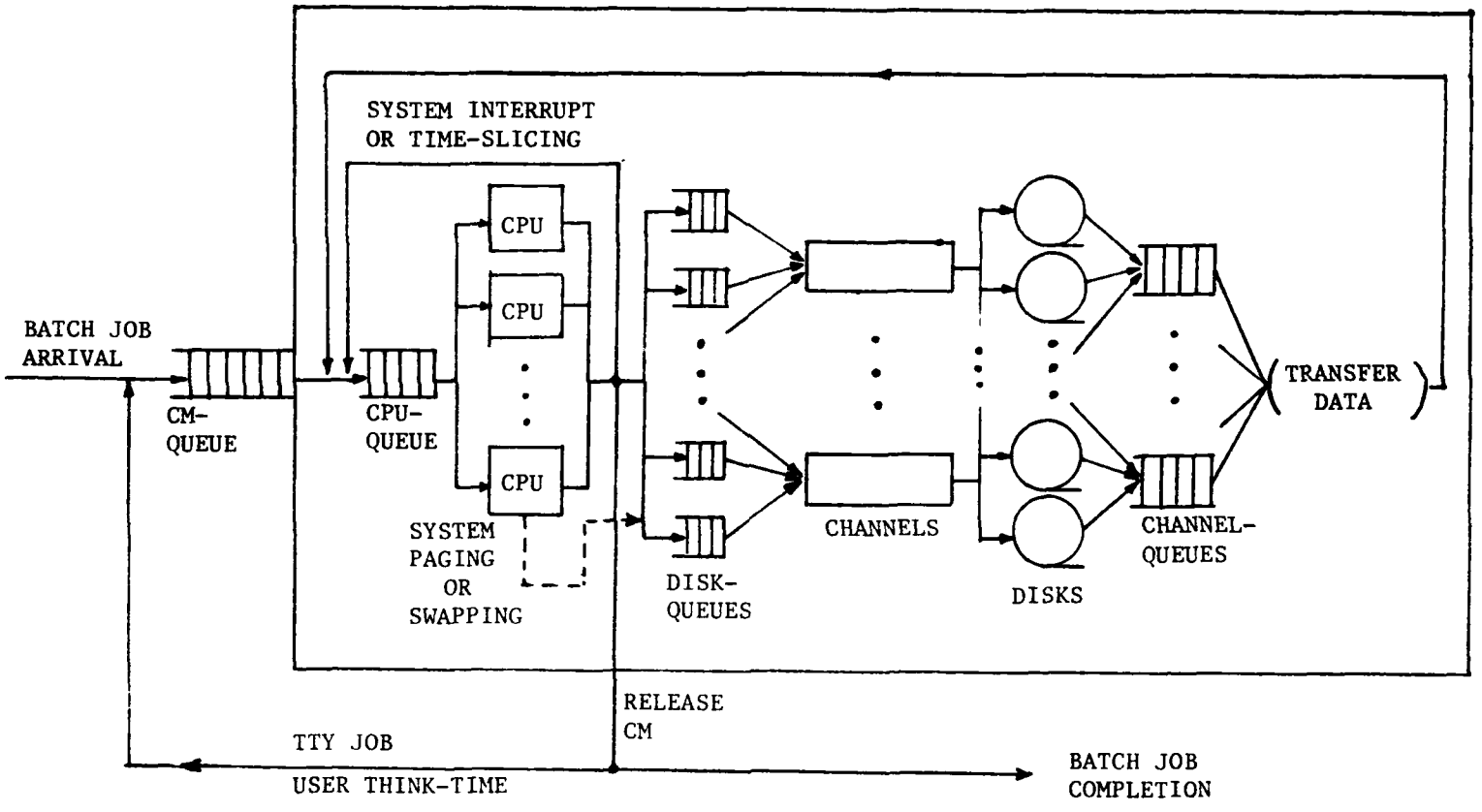
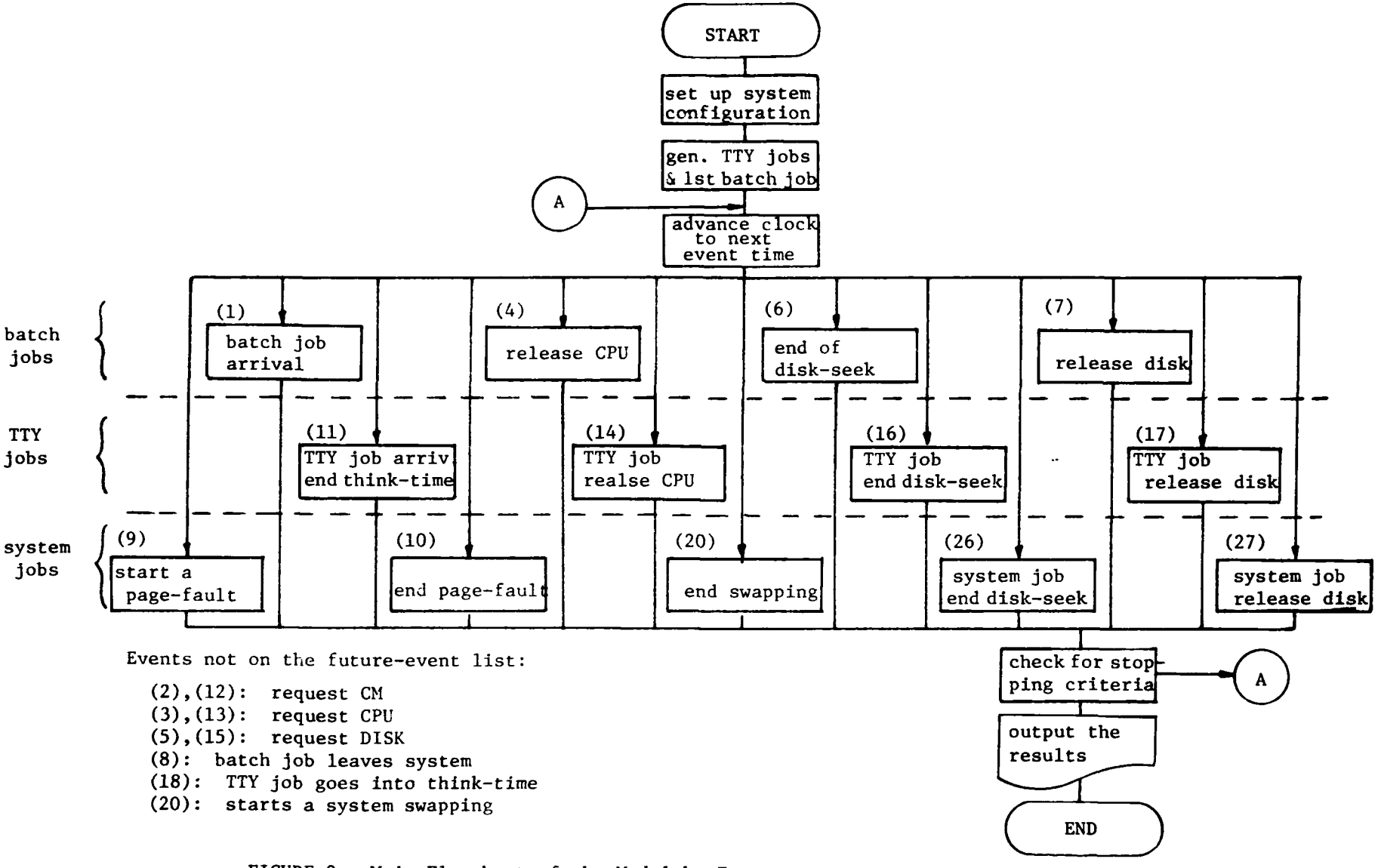


FIGURE 1. The Job-Flow of the Model



Events not on the future-event list:

(2),(12): request CM
(3),(13): request CPU
(5),(15): request DISK
(8): batch job leaves system
(18): TTY job goes into think-time
(20): starts a system swapping

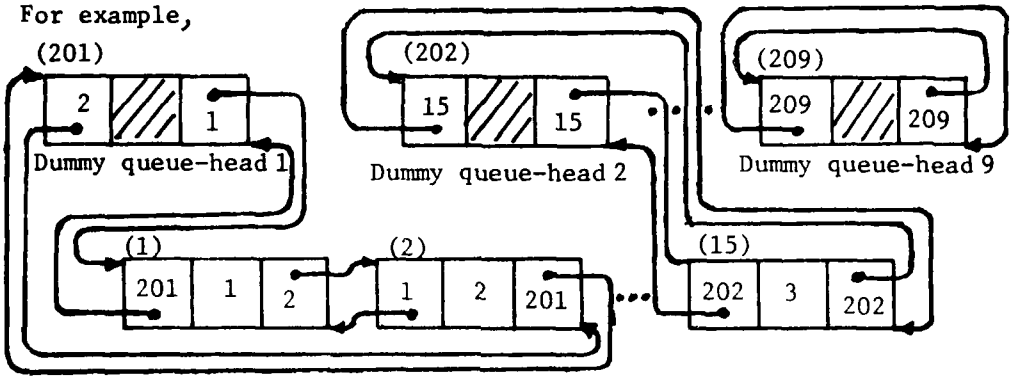
FIGURE 2. Main Flowchart of the Model by Events

A record node has the following format:

(pointer)

predecessor link	user-job number	successor link

where (pointer) is the record number



To put the above queues in the form of an array, we have:

	RECORD NO.	PRE	JOB NO.	SUC
Record nodes	1	201	1	2
	2	1	2	201
	15	202	3	202
dummy queue-heads	201	2		1
	202	15		15
	209	209		209

FIGURE 3. Doubly-linked queue structure in the model.

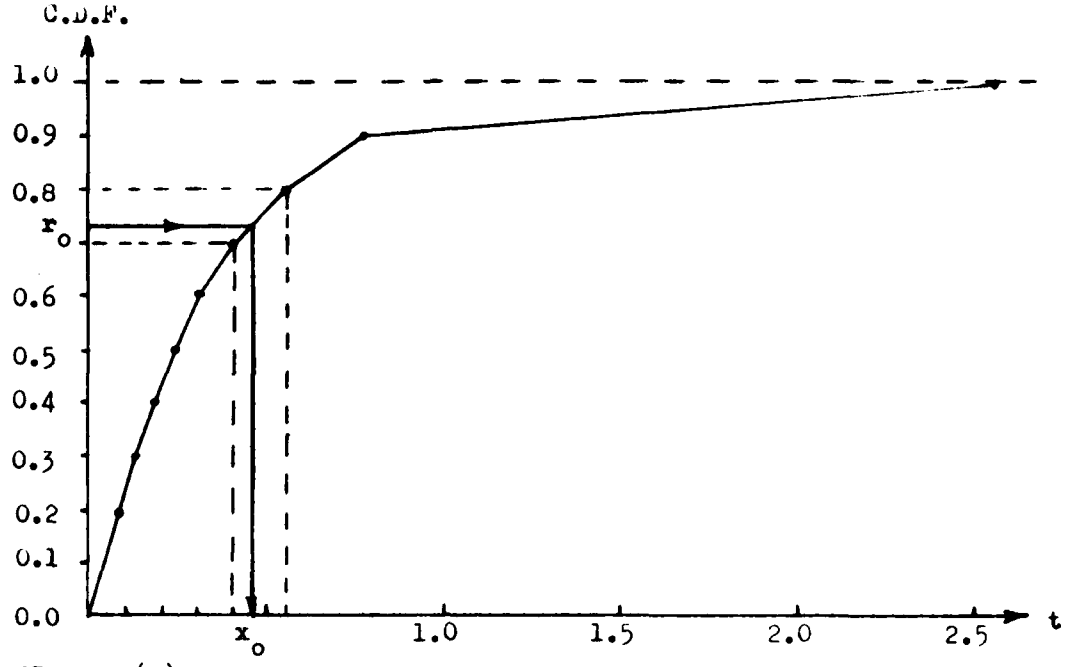


FIGURE 4(a) Generation of a random variate x_o from a discrete C.D.F.. (r_o is a random number uniformly generated between 0 and 1.)

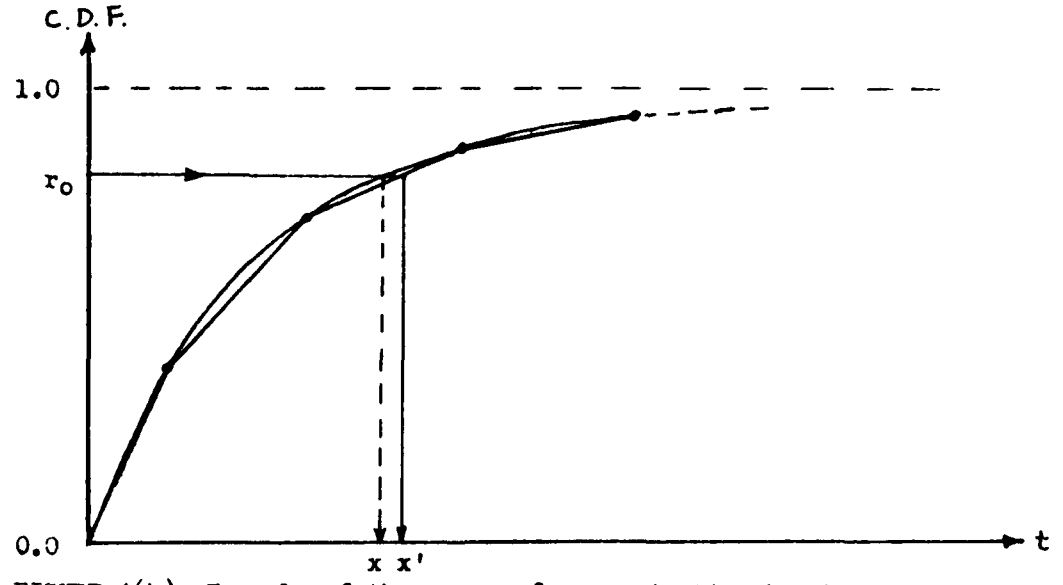


FIGURE 4(b) Example of the error of approximation by the discrete C.D.F.. (x' is the estimation of x.)

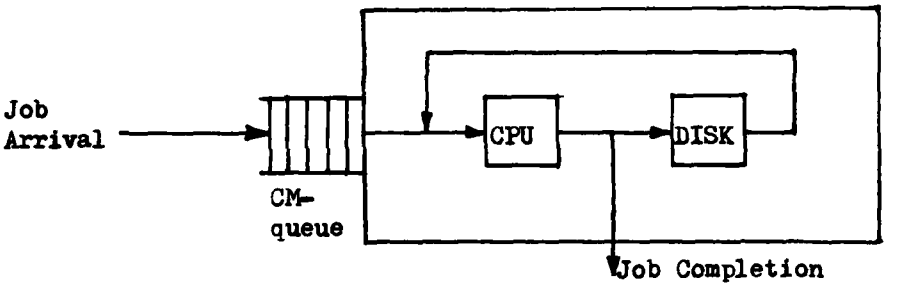


FIGURE 5 A simple 1-CPU, 1-disk uniprogramming system.

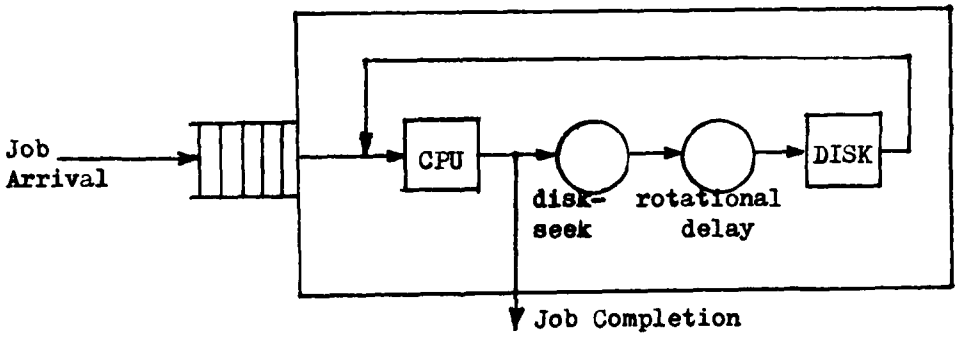


FIGURE 6 A simple system with disk-seek and latency time.

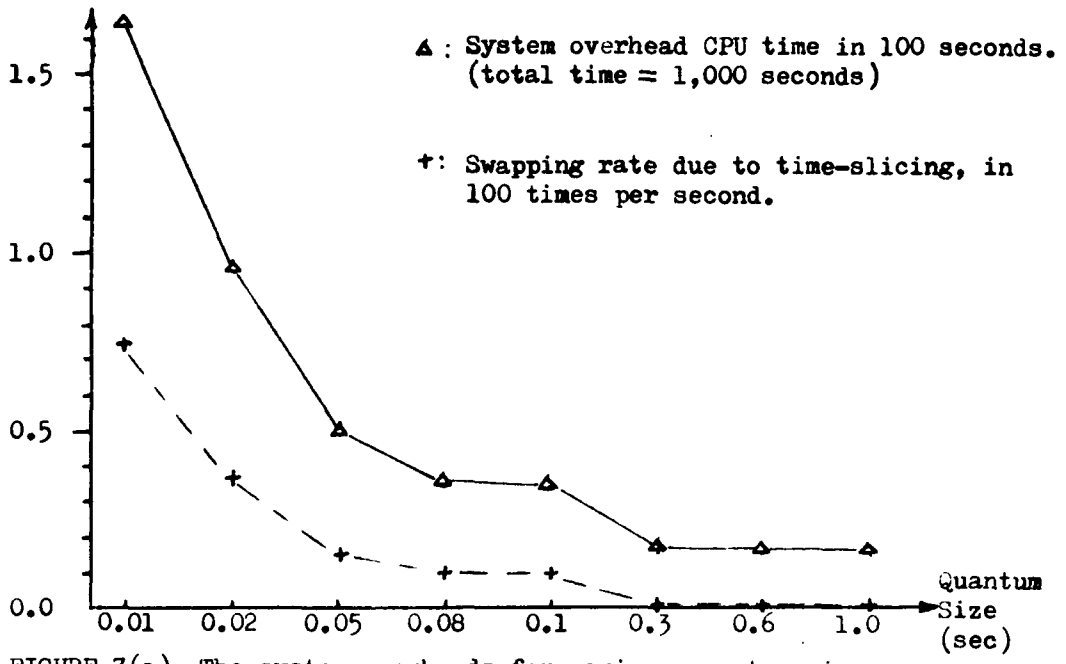


FIGURE 7(a) The system overheads for various quantum sizes.

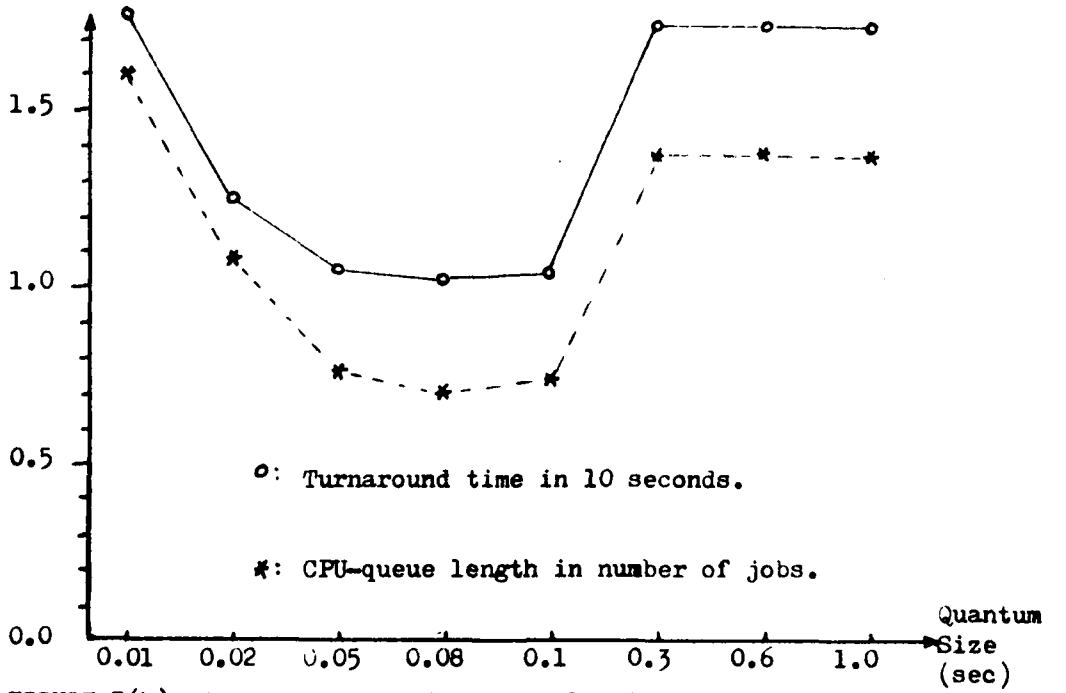


FIGURE 7(b) Turnaround and CPU-queue length for various quantum sizes.

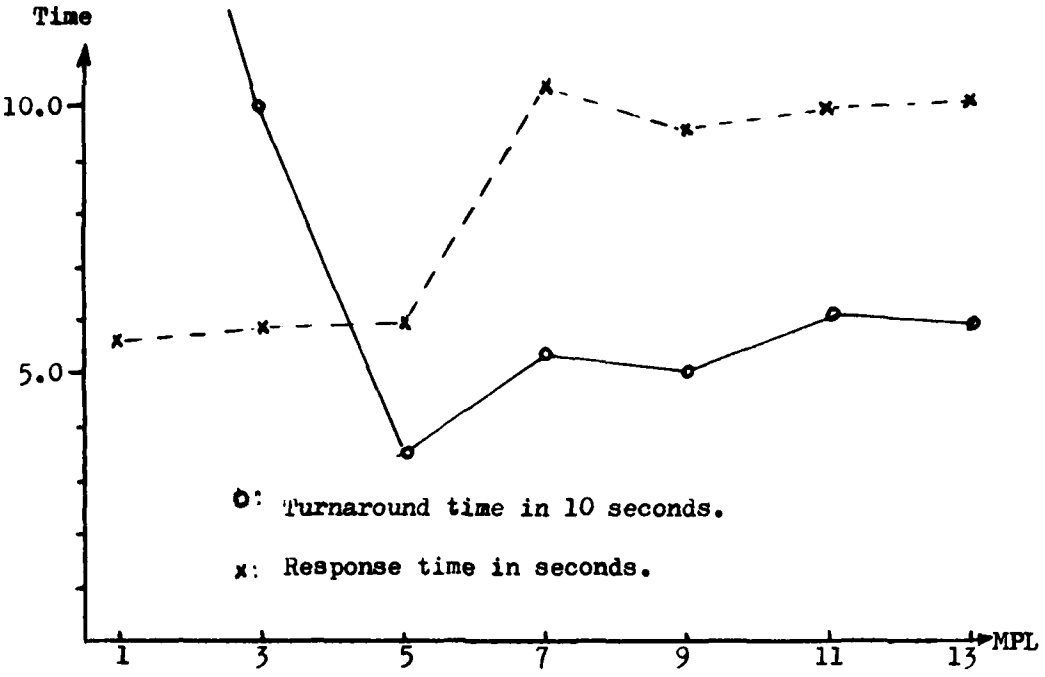


FIGURE 8(a) Turnaround and response time under different Multi-Programming Levels.

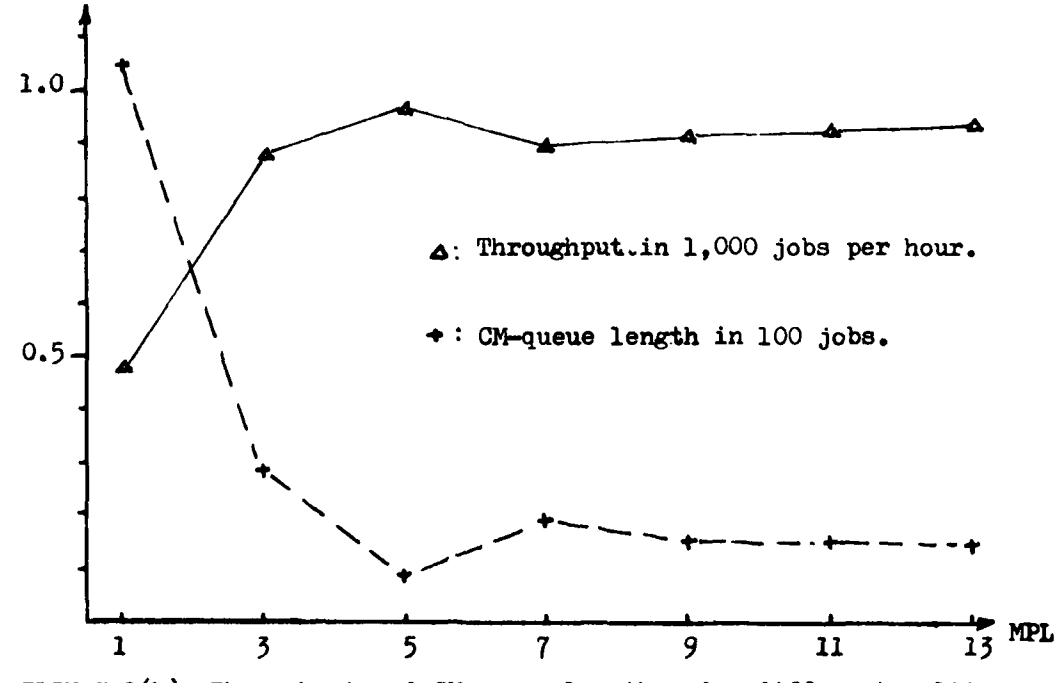


FIGURE 8(b) Throughput and CM-queue length under different Multi-Programming Levels.

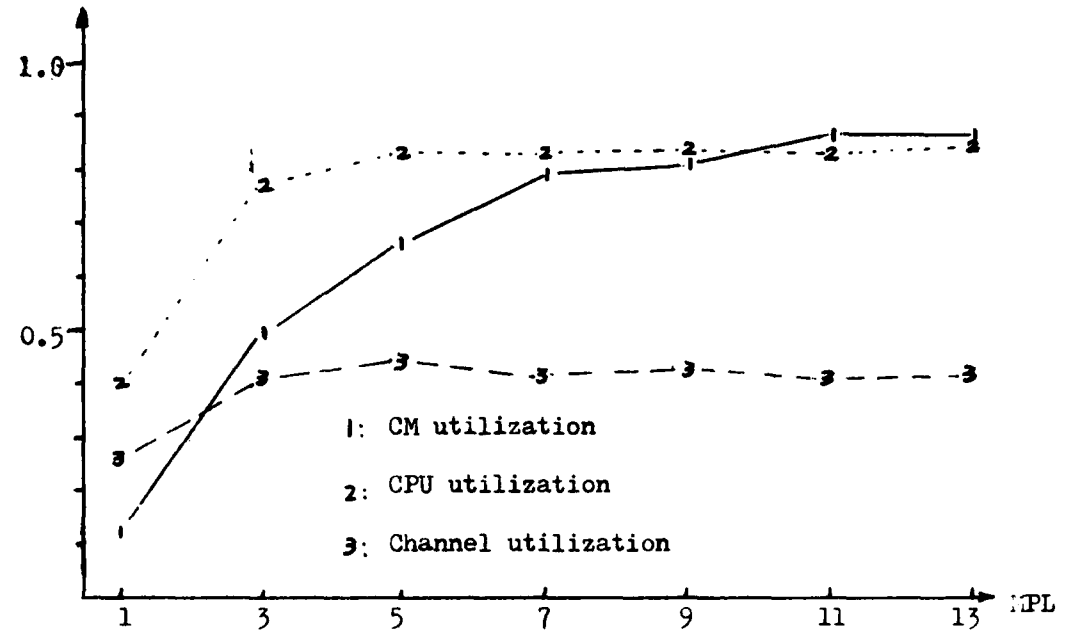


FIGURE 8(c) System utilizations under different Multi-Programming Levels.

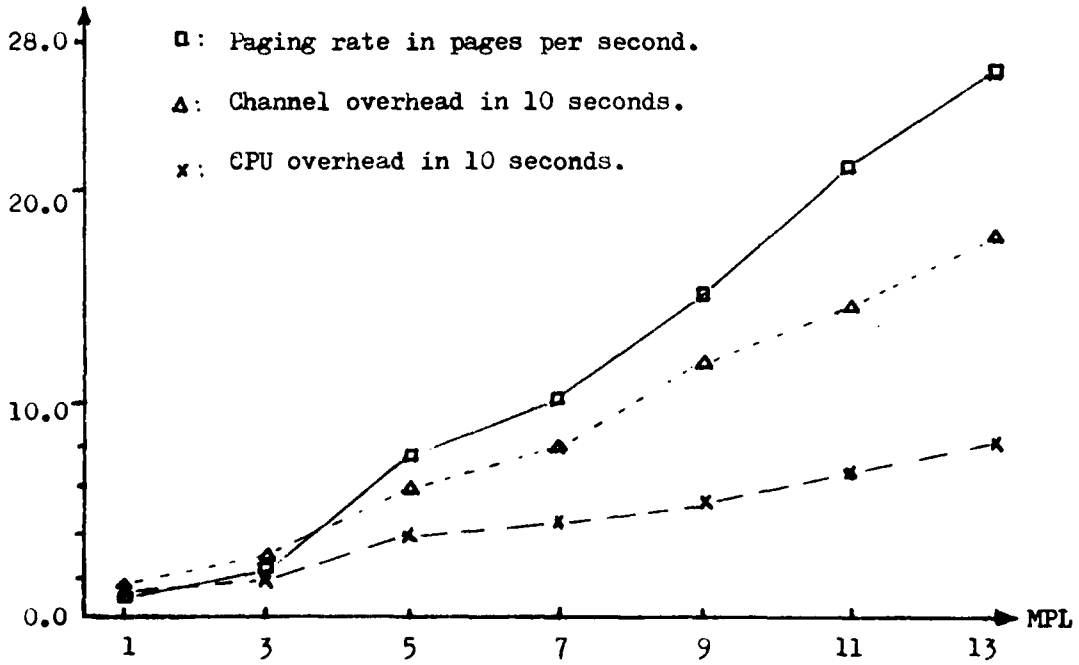


FIGURE 8(d) System overheads under different Multi-Programming Levels.

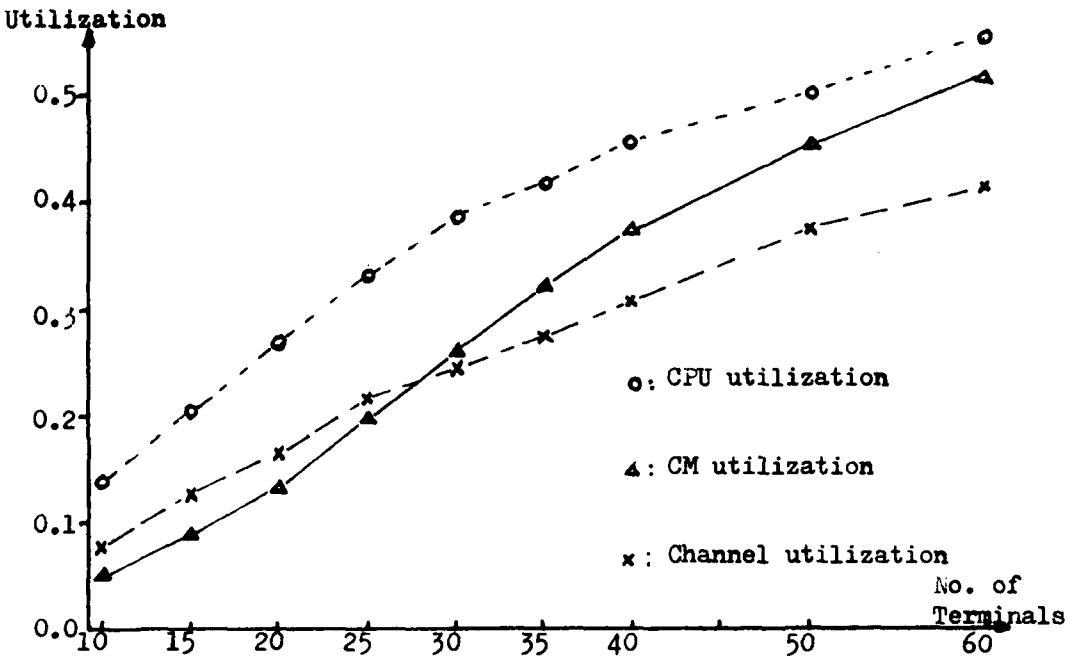


FIGURE 9(a) System utilizations vs. number of terminals.

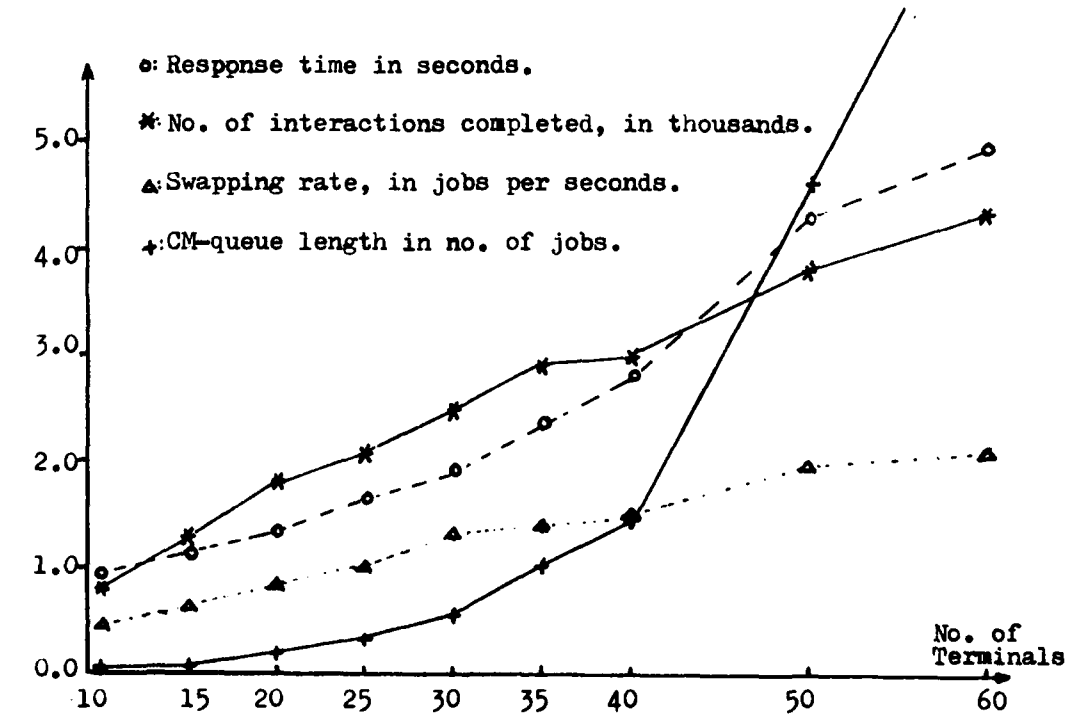


FIGURE 9(b) System overheads and response time vs. number of terminals.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER **AFOSR-TR- 80-0339** 2. GOVT ACCESSION NO. AD-A084 405	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A HIGHLY PARAMETERIZED TOOL FOR STUDYING PERFORMANCE OF COMPUTER SYSTEMS	5. TYPE OF REPORT & PERIOD COVERED INTERIM 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Herman D. Hughes	8. CONTRACT OR GRANT NUMBER(s) AFOSR 78-3547
9. PERFORMING ORGANIZATION NAME AND ADDRESS Michigan State University Department of Computer Science East Lansing, MI 48824	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61102F 2304/A2
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Office of Scientific Research/NM Bolling AFB, Washington, DC 20332	12. REPORT DATE March 1980 13. NUMBER OF PAGES 24
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

simulation model, queue, scheduling policies, workloads, hardware configuration, model validation, system performance, events, cumulative distribution function.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A highly parameterized simulation model is described which allows experiments to be performed for computer performance evaluation studies. The results of these experiments can be used to evaluate the effect of changing the hardware configuration, the workload, the scheduling policy, the multiprogramming level, etc. The model is constructed tofunction either as a batch or time-sharing system, or as a combination of both. This simulation model also has the potential of providing dynamic feedback for the scheduler. A discussion of the design, implementation, and use of the model is presented. Examples are provided to

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

20. Abstract cont.

illustrate some possible uses of the model and verifications of the results obtained from the model.